(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 798 655 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
01.10.1997 Bulletin 1997/40

(51) Int Cl.[6]: **G06F 17/30**

(21) Application number: **97301684.3**

(22) Date of filing: **13.03.1997**

(84) Designated Contracting States:
DE FR GB IT NL

(30) Priority: **26.03.1996 US 621578**

(71) Applicant: **SUN MICROSYSTEMS, INC.**
Mountain View, California 94043-1100 (US)

(72) Inventors:
- **Jervis, Robert B.**
  Monte Sereno, California 95030 (US)
- **Nevin, Thomas D.**
  Fremont, California 94536 (US)
- **Foley, Jill Paula**
  San Jose, California 95117 (US)
- **Sielski, Karen Lynn**
  Sunnyvale, California 94087 (US)

(74) Representative:
Cross, Rupert Edward Blount et al
BOULT WADE TENNANT,
27 Furnival Street
London EC4A 1PQ (GB)

(54) Internet-enabled graphical user interface with toolbar icons having built-in links to world-wide web documents and an integrated web browser

(57) A graphical user interface is disclosed that consists of a toolbar and toolbar icons that can be directly linked to Web documents containing references to Java applets or data files that collectively implement the functions associated with the icons. The Web documents and/or the referenced objects can be remote (i.e., accessible over the Internet) or local (directly accessible via the operating system of the computer displaying the GUI). The graphical user interface is linked to Web browser software so that, whenever a toolbar icon is selected, the interface triggers the Web browser to load the Web document linked to the selected icon. Once this occurs, the Web browser automatically loads the files referenced by the Web document and then executes those of the loaded files that are executable (i.e., Java applets or standalone programs). In a preferred embodiment, each toolbar icon is linked to a single applet (remote or local) that implements the icon's functions. Due to the linkage between an icon's associated Web document and the Web browser, the icon's associated applet is executed automatically whenever the icon is selected.

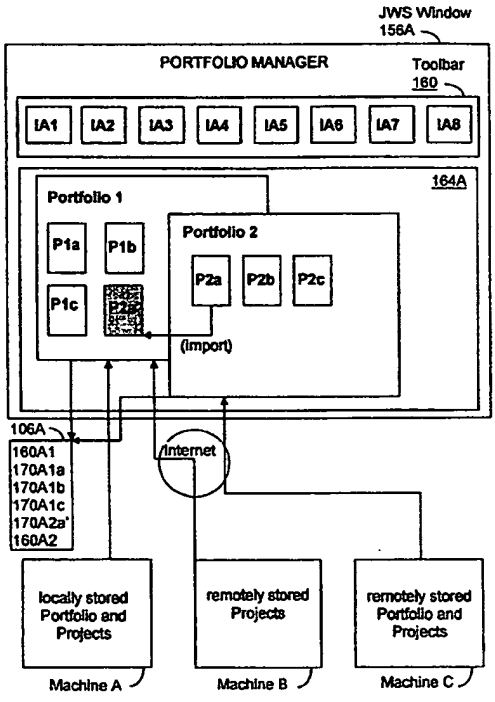

FIG. 6

EP 0 798 655 A2

Description

The present invention relates generally to graphical user interfaces and particularly to the design of a graphical user interface wherein icons in a toolbar are linked to respective Web documents which are loaded and executed when their linked icon is selected.

BACKGROUND OF THE INVENTION

Graphical user interfaces (sometimes referred to as GUIs) are well known mechanisms by which users can interact with computer programs and files. The typical GUI provides a set of selectable icons, each associated with a particular operation provided by the program controlling the GUI or a file that can be accessed from the controlling program. A user wanting to initiate an operation from the GUI does so by selecting (e.g., with a mouse) the appropriate icon. For example, a user of a word processor can initiate spell checking on the active document by selecting the spell checking icon from the word processor's toolbar. Similarly, a user can move a file from one directory to another by selecting the file's icon, dragging the selected icon to another icon representing the destination directory and then dropping (deselecting) the icon. This type of GUI design is also well-suited to launching standalone programs (e.g, by double-clicking on the icon representing an executable program).

In each of these cases, the immediate operations initiated by an icon's selection must be local to the network or computer hosting the operating system or application program that displayed the selected icon. This means, for example, that an icon cannot be directly associated with an executable program located on a remote system. Instead, in the prior art, the execution of a remote program can only proceed indirectly, by linking the icon for the remote program to a local executable program that establishes communications with the remote system and then downloads and executes the remote program on the local system. Of course, this scheme only works when the local program and the remote system support compatible communications modes and the remote program has been pre-compiled so that it can be executed on the local system.

Therefore, there is a new for a graphical user interface that allows icons to be linked to remote objects, such as programs or files, so that the remote objects can be downloaded and/or executed using the services of the operating system or application displaying the icon. There is also a need for such a GUI to be associated with a host system that guarantees that the local services that handle icon selection events and the remote systems on which remote objects are stored are able to communicate. Additionally, it would be desirable for a subset of all executable, remote files to be executable on any system with which the remote system is guaranteed to be able to communicate.

Some of these features are embodied in the World-Wide Web ("WWW"), which links many of the servers making up the Internet, each storing documents identified by unique universal resource locators (URLs). Many of the documents stored on Web servers are written in a standard document description language called HTML (hypertext markup language). Using HTML, a designer of Web documents can associate hypertext links or annotations with specific words or phrases in a document (these hypertext links identify the URLs of other Web documents or other parts of the same document providing information related to the words or phrases) and specify visual aspects and the content of a Web page.

A user accesses documents stored on the WWW using a Web browser (a computer program designed to display HTML documents and communicate with Web servers) running on a Web client connected to the Internet. Typically, this is done by the user selecting a hypertext link (typically displayed by the Web browser as a highlighted word or phrase) within a document being viewed with the Web browser. The Web browser then issues a HTTP (hypertext transfer protocol) request for the requested document to the Web server identified by the requested document's URL. In response, the designated Web server returns the requested document to the Web browser, also using the HTTP.

The standard HTML syntax of Web pages and the standard communications protocol (HTTP) supported by the WWW guarantee that any Web browser can communicate with any Web server. However, until the invention of the Java programming language and Java applets, there was no way to provide platform-independent application programs over the Interact and the WWW.

Important features of the Java programming language include the architecture-independence of programs written in the Java language, meaning that they can be executed on any computer platform having a Java interpreter, and the verifiability of the integrity of such programs, meaning that the integrity of Java programs can be verified prior to their execution. A Java program verifier determines whether the program conforms to predefined stack usage and data usage restrictions that ensure that verified programs cannot overflow or underflow the executing computer's operand stack and that all program instructions utilize only data of know data types.

As a result, Java language programs cannot create object pointers and generally cannot access system resources other than those resources which the user explicitly grants it permission to use. Consequently, when one or more code fragments are downloaded to a client along with an associated form or image file, a Java-compatible browser running on the client will be able to verify and execute the downloaded code fragments needed to display the image or fill out

the forms.

Thus, Java-compatible Web browsers are able to download from any WWW server Java applets that are guaranteed to be executable on the local system. However, existing Web browsers are configured to download Web documents only in response to a user selecting a hyperlink in a Web page being currently displayed by the browser or a user entering a URL into the browser. Web browsers are not configured to provide these Web-related services in response to a user selecting from a GUI an icon that is associated with remote objects, either executable programs (standalone programs or applets) or other files. Thus, there is a need for a GUI that provides icons that can be associated with local or remote files and programs. Then, when an icon associated with a remote file or program is selected, this GUI should invoke local web browser services on the URL of the remote file or program, resulting in the file or program and all other objects referenced in such a file or program being downloaded to the local machine and executed, if possible.

SUMMARY OF THE INVENTION

The present invention is a computer-readable memory configured to direct a computer networked with a set of remote computers to display a set of elements of a graphical user interface and initiate operations indicated by selection of one or more of the displayed set of elements. This computer memory includes browser software and a toolbar specification that defines characteristics associated with the set of elements. More particularly, the toolbar specification defines for each element in the set of elements a set of visual attributes and a set of links to related files.

The set of visual attributes defines the appearance of an element as displayed by the computer. Each link in a set of links associates an element with a document that includes references to files needed to implement an operation associated with the element. The referenced files can have a type selected from an applet or a data file and can be located on the computer or one of the remote computers. Consequently, a link can be selected from a local link to files stored on the computer or a network link to files stored on the remote computers. Given these data structures, when one of the elements is selected, the browser directs the computer to load files referenced in the documents associated with the selected element via the set of links and execute any of the loaded files that are executable.

BRIEF DESCRIPTION OF THE DRAWINGS

Examples of the invention will be described in conjunction with the drawings, in which:

Figure 1 is a block diagram of a computer network showing details of the memory and display associated with one of the networked computers.

Figure 2 is a data flow diagram illustrating the processing steps performed by the present invention following the selection of an icon from the toolbar.

Figure 3 is a figure showing the data structures employed in a preferred embodiment of the Java Workshop.

Figure 4 shows instances of a portfolio manager menu and a submenu associated with one of the portfolio manager methods.

Figure 5 shows the structure of a portfolio file that is employed in a preferred embodiment of the Java Workshop.

Figure 6 is a depiction of a display window generated by the portfolio manager of the present invention where some of the displayed components are local or remote to the computer displaying the portfolio manager window.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to Figure 1, there is shown a computer network 100 with at least three computers: A 102A, B 102B and C 102C. Each computer 102 includes a processor 104, a memory 106, which could be a fast, primary memory (e.g., a RAM) or a slower, secondary memory (e.g., a hard disk drive), and a display 108. The computers 102 operate in accordance with well known computing principles (i.e., each computer 102 executes programs in its memory 106 under the control of an operating system (not shown), which provides system services for the executing programs). In the preferred embodiment. the interconnections 103 between the computers 102 are provided by the Internet, although the present invention is also applicable to any environment wherein networked computers can communicate using a standard communications protocol (such as HTTP) and wherein platform independent programs can be downloaded and executed over the network from within browser software. For the purposes of explaining the operation of the present invention, it is assumed that there is no network operating system that coordinates file exchange operations between the three computers A, B and C.

Details of the present invention are now described in relation to the particular implementation shown in Figure 1. In this implementation, the user interface of the present invention is embedded within an application called the Java Workshop (JWS) program 150A, which, among other things, allows users to organize executable programs (Java applets and standalone executables) and non-executable files (image files and Java class libraries) into collections called portfolios. In a major departure from the prior art in the area of program and file managers, the JWS program

150A has an integrated JWS Browser 154A that allows a user seamlessly to create and work with portfolios that are remote (stored apart from the user's machine or local network) or local. Furthermore, the JWS browser 154A allows portfolios to be assembled that are mixtures of local and remote "projects".

The term "projects" is defined for the purposes of this document to mean the components of a portfolio.

The user interface of the present invention facilitates user interaction with mixed objects, such as a portfolio consisting of both local and remote projects, by providing a single paradigm for working with all objects regardless of the objects' locations. Of course, there are differences between working with remote and local objects. For example, executing a Java applet stored on a remote computer is a vastly different task from executing a standalone program stored locally. These differences are handled in the JWS program 150A. However, the user interface of the present invention allows a user to initiate the execution of the remote applet or the local program in the same way (e.g, by double-clicking an icon representing the applet). How the user interface of the present invention provides this location-tranparency flexibility is now described in reference to Figure 1.

Referring to Figure 1, the memory 106A includes a set of JWS files 110A that collectively define the user interface, methods and data files that compose the Java Workshop (JWS). More specifically, the JWS files 110A include the JWS program 150 (hereinafter referred to as the "JWS"), JWS browser 154A and a group of interface files called the JWS toolbar specification 112A. The JWS toolbar specification 112A is composed of four sub-groups of files: icon specifications 114A, web documents 120A, JWS applets 140A and other referenced files 148A. The elements 114A, 120A, 140A, 148A specify the appearance and, more importantly, the operation of a set of icons (IAi) 162Ai that are displayed on the display 108A as elements of a JWS toolbar 160. The JWS toolbar 160, which is a key element of the JWS user interface, is displayed by the JWS program 150A on the JWS window 156A. The JWS window 156A also includes an applet window 164A that is controlled by JWS applets 140A that are executed by the JWS program 150A in the course of project and/or portfolio management.

Each icon IAi has a corresponding icon specification 114A that defines the icon's visual attributes 116Ai and specifies a link(s) 118Ai to a Web document(s) 120Ai that lists an initial set of files that are to be load and, possibly, executed whenever the icon IAi is selected. The links 118Ai can be to Web documents 120Ai that are stored on the local system (e.g, the computer 102A), in which case a link comprises a local path and file name that can be handled by the file service provided by the local operating system (not shown). The links 118Ai can also be to remote Web documents (e.g., documents stored on the computers 102B, 102C) that can be retrieved over the Internet by a conventional Web browser. Because the JWS program 150A incorporates a JWS Browser 154A that provides all of the features of a conventional Web browser, it does not matter where the Web document(s) 120Ai linked to a particular icon IAi are stored, nor does it matter on what type of platform the linked documents are stored. What is important to the preferred embodiment is that the JWS browser 154A is able to communicate with the remote platform hosting the Web document 120A1 via one of the standard communications protocols supported by the Internet, such as HTTP or FTP. If this is the case, the linked Web documents 120Ai will be automatically downloaded by the JWS browser 154A (triggered by the JWS program 150A) whenever their corresponding icon is selected. This eliminates many of the complexities that would be required in the prior art to implement a similar feature linking icons to remote, executable documents.

Each Web document 120A (which could have initially been stored locally or remotely before being loaded into the memory 106A) includes two elements: a title 122Ai and a set of references to its components 124Ai. A Web document 120A can also include embedded files (not shown); however, as Web browsers do not make a functional distinction between embedded and referenced files, neither will this application. As with the links 118Ai, the references 124Ai in the Web documents 120A can be to remote or local files. In either case, they are handled by the JWS browser 154A in the same manner as described for the links 118Ai. One significant advantage of the present user interface is that a reference 124Ai can be to a Java applet 140Ai that is responsible for handling the operations associated with the icon IAi whose related Web document referenced the applet 140Ai. In this situation, when the JWS browser 154A retrieves the web document 120Ai linked to a selected Icon IAi, it automatically will pull in and begin executing the referenced applet 140A (which could have been stored on a remote system). The applet 140Ai, running in the JWS browser's virtual machine, can then implement the icon's associated operations without needing to worry about network and operating system complexities, which are handled respectively by the local operating system and the JWS browser 154A.

In the preferred embodiment, a single JWS applet 140Ai is referenced in each Web document 120Ai. This single applet controls or directly implements all of the functions associated with one icon IAi. For example, in the preferred embodiment, a spell checker icon IA1 could be linked via a Web document 120A1 to a remote applet 140A1 that, once downloaded to the computer 102A and executed by the JWS browser 154A, spell-checks the appropriate document (s). Alternatively, a Web document 120Ai can reference many applets 140Ai. For example an icon IA2 could be linked to a Web document 120A2 that references a spell-checker applet and a grammarchecker applet so that both are automatically brought up by the JWS browser 154A whenever the icon IA2 is selected from the toolbar 160. In addition to applets, a Web document of the present embodiment can reference other types of components 148A, including data and image files.

Referring to Figure 2, there is shown a data flow diagram illustrating the series of steps by which an applet is invoked in response to the selection of a particular icon IA1 from the toolbar160. Each icon selection event is monitored by the JWS browser 154A, which, following the selection of the icon IA1, retrieves the link 118A1 from the icon IAI's specification file 114A1. Via the link 118A1, the icon IA1 is associated with the Web document 120A1, which is automatically loaded by the JWS browser 154A. The JWS browser 154A then loads all of the files referenced in the document 120A1 and also executes any of the referenced files that are executable (i.e., the applets). In this example, it is assumed that there is one referenced executable, the applet 140A1. Once it is active, the applet 140A1 can take control of a portion of the display 108A (e.g., the applet window 164A), on which it can display results, dialog boxes and icons that facilitate user interaction with the applet's functions and capabilities.

Referring to Figure 3, there is shown a data structure diagram setting out additional details of data items stored in the memory 106A that are used by the Java WorkShop Program 150A. These data items include icon specifications 114Ai that define the visual attributes 116Ai and Web document links 118Ai for icons IAi displayed by the JWS 150A on the toolbar 160. These icons (shown on Fig. 2) include a portfolio manager icon IA1, project manager icon IA2, text edit project icon IA3, build/compile icon IA4, source browse icon IA5, debug icon IA6, run icon IA7 and help icon IA8. When selected, the icons IA1-IA8 respectively allow a user to access the following capabilities (and menus) provided by the Java Workshop 150A for working with JWS portfolios and projects:

IA1: provides access to the portfolio manager applet 140A1, which displays the projects in the current portfolio;
IA2: provides access to the project manager applet 140A2, in which the user can edit project information;
IA3: provides access to a JWS text editor applet (not shown) in which the user can edit project source code;
IA4: provides access to a JWS project compiler applet (not shown);
IA5: provides access to a JWS browser applet (not shown) that allows the user to browse JWS source programs included in the other referenced files 148A;
IA6: provides access to a JWS debugger applet (not shown) that allows the user to debug JWS source programs included in the other referenced files 148A;
IA7: provides acess to a JWS project run method 146A2f that runs executable projects (i.e., applets and standalone programs); and
IA8: provides access to a JWS help applet (not shown) that provides context-sensitive help for JWS operations.

As described in reference to Figure 1, in the preferred embodiment, an icon specification 114Ai includes a link to a Web document 120Ai that has a reference 124Ai to a single applet 140Ai that implements the operations associated with the corresponding icon IAi. Thus, the icon specification 114A1, which is associated with the portfolio manager icon IA1, is linked to a Web document ("Portfolio.HTM") 120A1 that includes a single reference 118A1 to the portfolio manager applet 140A1. Similarly, the icon specification 114A2, which is associated with the project manager icon IA2, is linked to a Web document ("Project.HTW") 120A2 having a single reference 118A2 to the project manager applet 140A2. These applets 140A1, 140A2 provide methods 146A1, 146A2 that can be applied respectively to portfolios and projects.

The methods 146Ai are made available to users as options on menus 147Ai that are displayed when their associated applet's icon is selected. For example, the portfolio managers methods 146A1 are displayed as options on a "Portfolio" menu 147A1. In conventional GUI fashion, when one of the methods/options is subsequently selected from its parent menu, that option's submenu, or page, is then displayed by the JWS 150A and enabled for user interaction. Many of the submenus 147Aij provided by the preferred embodiment; eg., the Project->Create, Import, Choose, and Remove submenus 147A1a,147A1b,147A1c,147A1d and the Project->Create, Import, Choose, Remove, Run and Copy submenus 147A2a,147A2b, 147A2c, 147A2d, 147A2e, 147A2f respeclively provide a list of portfolios and projects to which the submenus associated method can be applied. For example, referring to Figure 4, there is shown an instance of the Portfolio→Choose submenu 147A1c listing a set of portfolios (Portfolio 1, Portfolio 2, Portfolio 3) for the Portfolio->Choose method 146A1c. Figure 4 also show the portfolio menu 147A1 listing the portfolio manager methods 146A1 (Create, Import, Choose, Remove).

Referring again to Figure 3, the methods 146Ai of the JWS applets 140Ai are now described from the vantage point of a user working with their respective menus 147Ai and submenus 147Aij. Most of this discussion focuses on the methods of the portfolio and project manager applets 114A1, 114A2, which are key elements of the JWS 150A.

Portfolio Manager Methods

The portfolio manager applet 140A1 provides four methods 146A1 that respectively allow a user of the JWS 150A to "Create" 146A1a, 'Import" 146A1b, "Choose" 146A1c and "Remove" 146A1d portfolios. Each of these methods 146A1 accomplishes its respective task by interacting with a set of portfolio files 160Ai, each of which can be stored on the local or remote systems and represents one portfolio. As shown in Figure 3, a generic portfolio file 160A includes

a collection of references 162A1i to its portfolio's constituent project files 170A. As with other file references in the present invention, a project reference 162Ai can be to a locally-stored project, in which case the reference is a local file name ("Name"), or to a Web document, in which case the reference is a URL.

For example, referring to Figure 5, there is shown a portfolio file 160A1 that contains project file references 162A1j for its constituent projects, which include an "Applet", a "Standalone" program, a Java "Package", an "Image" and a "Remote" applet, all of which are local projects stored in the user's 'home' (i.e., local) directory in the memory 106A Because these projects are all stored in the user's "home" directory, they can be read and written by the user and their corresponding project files can be referenced by path and file name. For example, the reference 162A1a to the applet project file 170A1 is "/home/Applet.prj". The portfolio file 160A1 also includes a reference 162A1f (/lib/SemiRemote.prj) to a project file for a read-only project ("SemiRemote") stored in a library directory on machine A and a reference 162A1g (http://B.com/Internet.prj) to a project file 170B1 for a read-only project ("Internet") stored on machine B that can only be accessed over the Internet using the JWS browser 154A.

Referring again to Figure 3, in the preferred embodiment, each user has a personal portfolio (with a corresponding portfolio file 160Ai) that contains only the projects that belong to that user. When the JWS 150A is initially activated, it brings up the personal portfolio as the current or active, portfolio. Using the portfolio manager's "Choose" method/option 146A1c, the user can **choose** another portfolio 160Ai to be the current portfolio. A user does this by selecting from the Portfolio->Choose submenu 147A1c (this terminology designates the Choose submenu that is displayed by the JWS 150A following the user's selection of the Choose option from the Portfolio menu 147A1) listing all available portfolios, the desired portfolio's file name (if it is local) or URL (if it is remote). The user can then view the projects composing the current portfolio by selecting the portfolio manager icon IA1 from the JWS toolbar 160. In this and other situations described herein, the applet being executed displays its results and menus on the applet window 164A.

A user of the JWS 150A can **create** a new portfolio by selecting the portfolio manager's "Create" option and then entering the mane of the portfolio to be created. In response, the JWS 150A calls the Portfolio->Create method 146A1a, which creates the corresponding portfolio file 160A on the local system, displays its name in the toolbar 160A and adds the portfolio's name to the Choose and Remove submenus 147A1c, 147A1d. The newly-created portfolio has no projects, but the user can add projects in the Project->Create submenu 147A2a (described below) or import existing projects into the portfolio with the Project→Import menu item 147A2b (also described below). Once the new portfolio has been created, its creator can keep it private or can publish it on the Internet to be accessed by others.

A user can also **import** existing portfolios that are not currently in their Portfolio→Choose submenu 147A1c. To import such a portfolio, the user first selects the Import option listed on the Portfolio menu 147A1. This triggers the Portfolio->Import method 146A1b to bring up an import submenu with a name field in which the user enters the file name or URL of the portfolio to be imported and an import button that the user clicks when they have completed their entries. In response, the import method 147A1c adds the portfolio name to the Portfolio→Choose and Portfolio->Remove submenus 147A1c, 147A1d. The JWS 150A also changes the current portfolio to the imported portfolio. Once it is on the Portfolio->Choose submenu 147A1c, the imported portfolio can be worked with like any other portfolio.

A user can **remove** a portfolio by selecting the portfolio to be removed from the Portfolio→Remove submenu. In response, the JWS 154A calls the Portfolio->Remove method 146A1d, which removes the selected portfolio from the Choose and Remove submenus 147A1c, 147A1d, but does not delete the portfolio's corresponding portfolio file 160A. Because the portfolio file is not deleted from the user's system, the user can at any time import the portfolio using the Portfolio->Import option 146A1b.

Each project file included in a portfolio has a corresponding project file 170A that describes the project and contains the project's contents. More specifically, each project file 170A contains the following information:

(1) the name 172A of the project;

(2) the project type 174A (Java applet (APPLET), standalone program (STANDALONE), Java class library (PACKAGE), data file IMAGE), an imported copy of a remote project of one of these four previously described types, or a remote applet (REMOTE));

(3) project administration information 176A, including whether the source code for the project should be distributed 178A to others requesting the project over the Internet and project options 180A;

(4) the project contents 182A, which can include the actual project contents and/or a set of references to other project files 170Ai, enabling multiple levels of embedded projects; and

(5) a run page URL 184A (applicable only for applet projects), which is the URL of the HTML file that includes an applet tag for the applet project.

This information determines which of the project methods provided by the JWS 150A can be employed by a user on a particular project. These project methods 146A2 are now described.

Project Methods

The JWS 150A provides several methods for working with projects. These methods are made available to users as options on a Project menu 147A2. When one of these methodsloptions is selected, the JWS 150A displays a corresponding submenu 147A2j from which the user specifies additional details of the operation. The project methods 146A2 include: Create 146A2a, Import 146A2b, Choose 146A2c, Edit 146A2d, Remove 146A2e, Run 146A2f, Copy 146A2g, and Paste 146A2h.

These methods allow a user to work with existing projects (local or remote) or create new projects. In either case, projects always exist in the context of a portfolio. When a project is created, it becomes the current project in the current portfolio.

A user can **create** a Java applet project, a standalone program project, a Java package project, an image project or a remote project. To create any of these projects, the user first "Chooses" the portfolio with which the project is to be associated and selects the "Create" option from the Project menu 147A2, upon which the JWS 150A calls the Project->Create method 146A2a. This method 146A2a displays the Project→Create page 147A2a, on which the user selects the type of project they wish to create (eg., if the user wishes to create an applet, they click on an applet button displayed on the submenu). The user then specifies the name of the package to be created and the local directory in the memory 106A in which the package's corresponding project file 170Ai is to be stored. Once the user has specified the attributes for the project the Project→Create methods adds a reference 162Aij to the project's corresponding project file to the portfolio file 170Ai of the current portfolio 160Ai.

In some situations (when the project being created is an applet, standalone program or Java package) the user may also have access to source code for the newly-create project. In these situations, the user enters the file names of the corresponding source files on the Project→Create page 147A2a. The JWS 150A adds these source file names to a "Sources" list maintained in the memory 106A so the source files can be accessed by the user (e.g., for editing and compilation). The user also enters the name of the main file for the program (i.e., the file that contains the "main" routine) of which the newly created project is a part. When the project being created is a Java applet, it is possible that the Java applet is referenced in an HTML page so that, when the applet's reference is selected, the applet will be executed. The JWS 150A allows such relationships to be represented via a Run Page URL field in the Project-Create page in which the user optionally enters the name of the HTML page that executes the applet.

When the user is creating an image project, after Choosing "image" from the Project→Create submenu 147A2a, the user enters the name of the image project and the URL of the corresponding image file. The user can then optionally enter attributes associated with the image, such as:

(1) the image's alignment with respect to surrounding text (e.g., choosing "bottom" alignment causes a browser displaying the image to align the bottom of the image with the bottom of the text);
(2) whether the image is active (meaning that a person viewing the image can click on different regions of the image to produce different actions); and
(3) an optimal text string that can be displayed in lieu of the image by browsers that are not able to display the image.

Once the user has filled in this information for the image project they are creating, the user clicks on the "Apply" field of the Project→Create page 147A2a, upon which the Project→Create method 146A2a makes the newly created image project the current project and displays the image in the Applet window 164A. The Create method 146A2a also adds the image project name to the Choose, Edit and Remove submenus 147A2c, 147A2d, 147A2e in the Project menu 147A2 and adds the name of the corresponding project file 170Ai to the portfolio file 160Ai associated with the current portfolio.

A user of the JWS 150A can **import** any type of project into one of their personal portfolios. They do this by choosing the portfolio they wish to be the current portfolio, selecting the "Import" option from the Project menu 147A2 and then entering the name or URL of the project to be imported on the Project→Import page 147A2b that is displayed by the Project→Import method 146A2b. After entering the necessary information, the user clicks an 'import" button displayed on the import page 147A2b, upon which the import method 146A2b imports the designated project into the current portfolio and adds the project name/URL to the Project→Choose, Edit, Remove and Run submenus 147A2c, 147A2d, 147A2e and 147A2f. The import method 146A2b also adds the name of the imported project's project file 170Ai to the current portfolio if it is not already contained therein. The JWS 150A does not make the imported project the current project, but the JWS 150A will display the imported project if the user subsequently selects the Portfolio Manager icon IA1 from the toolbar 160.

The JWS 150A allows a user to **create a remote applet** project. The user does this by "Choosing" the current portfolio, selecting the "Create" option from the "Project" menu and clicking the "remote applet" button displayed on the Project→Create submenu 147A2a. The user then enters the name of the project and the URL of the HTML page that executes the applet. Once these fields have been completed, the user exits the Project->Create submenu 147A2a

by clicking on "Apply". The create method 146A2 then creates a project file 170Ai with empty contents 182Ai and a run page URL field 184Ai that is set to the URL of the HTML page that executes the applet. For example, referring to Figure 5, the remote project file 170A5 has a run page URL 184A5 set to the URL ("http://C.com/RunApplet2.htm") of the Web page ("RunApplet2.htm") that runs the remote applet "Applet2". The create method 146A2 also adds the name of the project file 170Ai to the current portfolio's portfolio file 160Ai. The JWS 150A then makes the imported project the current project, loads the Portfolio Manager 140A1 and selects the current project to be displayed by the Portfolio Manager 140A1. The JWS 150A then adds the imported applet project's name to the Choose, Edit, Remove, Run and Copy submenus 147A2c, 147A2d, 147A2e, 147A2f in the Project menu 147A2.

A user can then run the remote applet by selecting its name from the Project->Run submenu 147A2f or by loading the Portfolio Manager, selecting the remote project and then pressing the Run button IA7 on the toolbar 160.

The Project→Run method 146A2f then passes the URL of the Web page referenced in the run page URL field (e. g., 184A5) of the remote applet project file (170A5) to the Web browser 154A, which downloads the referenced Web page (http://C.com/RunApplet2.htm) and runs the remote applet (Applet2).

If a user does not specify a Run page URL 184Ai in an applet's project file 170Ai, that applet project can still be run using the Project->Run method 146A2f. In this situation, the Project→Run method 146A2f automatically generates a new Web page that contains an applet tag created with the project attributes and parameters entered by the user on the Edit Project 146A2d run folder. This automatically generated HTML page is loaded into the JWS 150A, which uses the browser 154A to run the applet project. This feature allows users to execute applets without having to know the HTML syntax for referencing applets.

The Copy method 146A2g of the Project Manager 140A2 allows a user of the JWS 150A to **copy** an applet into an HTML file that executes the applet without requiring the user to know the HTML syntax for referencing applets. The user does this by first selecting (single-clicking on) an applet project in the current portfolio and then selecting the Copy option from the Project menu 147A2. This set of actions causes the Copy method 146A2g to copy the contents 182Ai of the selected applet project to a clipboard (not shown) maintained by the JWS 150A. The user then selects the Text Editor Icon IA3 from the toolbar 160A, upon which the JWS 150A executes the Editor method 146A2d. The editor method 146A2d brings up a text editor containing an Edit menu 147A2d, which includes a list of editing options, including "Paste". The user selects the "Paste" option from the Edit menu 147A2d, upon which the **paste** method 142A2h pastes the contents of the clipboard (i.e., the applet being copied) into a new file. The user can then save the new file as an HTML file, which causes the JWS 150A to add to the saved HTML file the appropriate links to the copied applet. As with other new projects, the JWS 150A adds the file name of the new HTML file 170Ai to the current portfolio's portfolio file 160Ai. Alternatively, the user can simply drag the image of the applet to be copied onto the image of an HTML file they wish to include the applet. The JWS 150A will then copy the contents of the applet to the designated HTML file and add to the HTML file tags referencing the copied applet.

The Project→Edit method 146A2d also allows a user to **edit** projects of all types. The Edit method 146A2d can be invoked by a user of the JWS 150A in one of two ways. First, the user can click on the Edit Project icon IA3 displayed on the toolbar 160 to invoke editing (i.e., the editing method 146A2d) on the current project. Second, the user can select the name of the project to be edited from the Project→Edit submenu 147A2d. Once editing is selected for a designated project, the JWS Editor method 146A2d opens on edit page 147A2d that includes six folders in which the user can edit information for the designated project. These six folders and their associated information include:

General	information about the project, including name, type and source directory
Build	information needed to compile the project
Debug/Browse	Information needed to debug and browse source files
Run	information needed to execute an applet or standalone program in the JWS Browser;
Publish	information needed to allow the project to be copied by other users; and
Portfolio	Portfolio information needed to display the project in the Portfolio Manager.

The project→edit method 146A2d allows a user to edit fields in these six folders only where appropriate. To assist the user, the edit method 146A2d greys out inapplicable fields. Whether or not a field is applicable depends on the type of the project being edited and whether the project is local or remote. For example, the edit method 146A2d will not allow a user to edit fields in the Debug/BBrowse folder for a project that is not a source file. The information that can be entered by a user in the General, Build, Debug/Browse and Run folders is mostly conventional, so it is not described in great detail. However, what is unique about the editing information in these folders is that the JWS 150A allows users to provide information for remote as well as local projects identlied by file names or URLs. This allows a user to specify, for example, that the source code for a particular project to be debugged or browsed exists on some remote node. This is not possible in conventional program and file management systems.

Because the preferred embodiment of the JWS 150A allows a user to employ portfolios and projects from remote sources and to publish their own portfolios and projects for others' use, this embodiment also provides a way for creators

of a project to indicate certain attributes of a project that are relevant to publication of a project on the Internet. These publication attributes are contained in the Publish and Portfolio Folders, which include the following fields:

Portfolio Folder

Description	a brief description of the project that is displayed by the JWS Browser when the mouse is positioned over the project image in the Portfolio Manager;
Portfolio Image URL	the URL for the image file (GIF, JPEG, or other) that represents the project image in the portfolio (If no image file is specified, a default GIF file is used by the JWS 150A); and
Features	the general characteristics of the project, for example, whether the project is video, graphics or audio.

Publish Folder

Distribute source copies	a toggle field with two values (YES/NO) that controls whether the project's source files are copied when the project is copied from one portfolio to another (when this field is set to NO, the JWS only allows the corresponding project files 170Ai to be copied absent the contents 182A); and
Submitter Name, E-Mail and URL	The name, e-mail address and Web page of the person adding the project to the portfolio.

A user can change the current project (i.e., the project being worked with in the JWS 150A) in one of two ways, in the first, the user starts by selecting the Portfolio Manager icon IA1 from the JWS toolbar 160. This causes the JWS 150A to open a Portfolio display showing the projects of the current portfolio in the Applet Window 164A. The user then selects from the Porttolio display the project they want to be the current project. The JWS 150A makes the selected project the current project and displays the name of the current project on the JWS toolbar 160. Alternatively, the user can change the current project by choosing the project name from the Project→Choose submenu 147A2c

The JWS 150A allows a user to **remove** a project from a portfolio in one of the following ways. First, in the portfolio manager display in the applet window 164A, the user can select the project they wish to remove and then click a Remove icon (not shown) provided by the Portfolio Manger 140A1. Alternatively, they can choose the name of the project to be removed from the Project->Remove submenu 147A2e. In either case, once the user has indicated the project to be removed, the Project→Remove method 146A2e removes the project from the Choose, Edit, Remove and Copy submenus 147A2c, 147A2d, 147A2e, 147A2g of the Project Menu 147A2. Note that the Project->Remove method 146A2e does not delete the removed project's project file 170Ai. This ensures that the user can subsequently import the project at a later time (using the Project→Import method 146A2b) if required.

Referring to Figure 6, there is shown an illustration of the user interface of the JWS 150A that highlights the resulting advantages of the present invention. The eight icons IA1-IA8 of the JWS 150A are shown on the toolbar 160. Two portfolios (Portfolio 1 and Portfolio 2) are shown on the applet screen 164A, which is under control of the portfolio manager 140A1. These portfolios represent two of the different types of portfolios that can be accommodated in the JWS 150A. Portfolio 1 is a local portfolio with a portfolio file 160A1 that includes three local projects P1a, P1b, P1b with corresponding project files 170A1a, 170A1b, 1701c. The shaded project P2a' in Portfolio 1 is a remote portfolio that was imported from Portfolio 2 over the Internet by the JWS Browser 154A (Fig. 1) under control of the JWS 150A. This project P2a' is represented in the memory 106A by the project file 170A2a'. Even though Portfolio 1 is mixed, its projects all are manipulable in the same fashion in the JWS 150A.

Portfolio 2 is a remote portfolio whose components are also stored remotely. The JWS 150A accesses the components of Portfolio 2 over the Internet using the JWS browser 154A but displays Portfolio 2 in the same manner as Portfolio 1. Note that a local portfolio such as Portfolio 1 could also reference only remotely stored projects (e.g., projects stored on a machine B). This ability of the JWS 150A seamlessly to organize projects and portfolios that may be distributed over the Internet is due to the integration of the JWS 150A and the JWS Web browser 154A. The Internet-awareness of the present embodiment also enables users to publish their own portfolios so they can be accessed and used by others over the Internet.

Claims

1. In a first computer having a display and a memory, said first computer being networked with a set of remote computers, a system for initiating from a graphical user interface displayed on said first computer the loading and execution of compound documents whose components are not constrained to exist solely art said computer, said

system comprising:

a displayable toolbar having a set of selectable icons, each icon being associated with one or more operations that are initiated via selection of said icon;
a set of compound documents, each of which includes a set of references to components, or files, that are needed to carry out one or more of said associated operations;
each of a subset of said files having a file type selected from an applet or a data file and a location selected from local or remote;
a set of links, each of which associates one of said icons with one or more of said set of documents, each of said links being selected from a local link when said document is stored on said computer or a network link when said document is stored on said remote computers; and
a browser that is configured, when one of said icons is selected, to cause said computer to load said files referenced in said documents associated with the selected icon via said set of links and execute any of the loaded files that are executable, thereby initiating the operation associated with said icon.

2. The system of claim 1, wherein said applets are written in a platform independent computer language that is interpreted and executed in a virtual machine implemented by said browser.

3. The system of claim 2, wherein said platform independent computer language is Java.

4. The system of claim 3, wherein said compound document is an HTML document and wherein each of said references is selected from a file name when the referenced file's location is local or a URL when the referenced file's location is remote.

5. The system of claim 4, wherein said HTML document includes a single reference to an applet that coordinates all of said operations associated with said icon; such that, when said icon linked with said HTML document is selected, said browser loads and executes said applet, which thereby coordinates and controls all operations and user and system interactions associated with the selected icon.

6. The system of claim 5, further comprising:
an applet specification file associated with one applet, said applet specification file indicating a set of input files on which its associated applet is to operate, said input files not being constrained to be located solely on said first computer.

7. The system of claim 6, wherein each of said input files stored remotely to said first computer is referenced by a URL in said applet specification file and each of said input files stored on said first computer is referenced by a local path and file name.

8. A method for initiating from a graphical user interface displayed on a first computer the loading and execution of compound documents whose components are not constrained to exist solely on said first computer, said first computer being networked with a set of remote computers that can host a subset of said components, said method comprising the steps of:

displaying a toolbar having a set of selectable icons, each icon being associated with one or more operations that are initiated via selection of said icon and being linked to one or more compound documents, each of which includes a set of references to components, or files, that are needed to carry out one or more of said associated operations, each of a subset of said files having a file type selected from an applet or a data file;
upon one of said icons being selected, loading said one or more compound documents linked to the selected icon;
upon loading said one or more compound documents, loading said files referenced in said compound documents, said files not being constrained to exist solely on said first computer; and
when a loaded file has a type that is selected from an applet or a standalone application, executing said loaded file.

9. The method of claim 8, wherein said applets are written in a platform independent computer language that is interpreted and executed in a virtual machine implemented by said browser.

10. The method of claim 8, wherein said step of loading said one or more compound documents comprises the steps of:

when a compound document to be loaded is stored on said first computer, retrieving said compound document using a local path and file name provided for said compound document in said link; and
when a compound document to be loaded is stored on one of said remote computers, issuing a document request message for said compound document over said network using a network node and file name provided for said compound document in said link.

11. The method of claim 10, wherein said step of loading said files comprises the steps of:

when a referenced file is stored on said first computer, retrieving said referenced file using a local path and file name provided for said referenced file in said reference; and
when a referenced file is stored on one of said remote computers, issuing a document request message for said referenced file over said network using a network node and file name provided for said referenced file in said reference.

12. The method of claim 11, wherein said compound document is an HTML document and wherein each of said references is selected from a local reference when said referenced file is stored on said first computer or a universal resource locator (URL) when said referenced file is stored on one of said set of remote computers.

13. The method of claim 12, wherein:

said HTML document includes a single reference to an applet that coordinates all of said operations associated with said icon; and
said steps of loading and executing comprise loading and executing said applet, which thereby coordinates and controls all operations and user and system interactions associated with the selected icon.

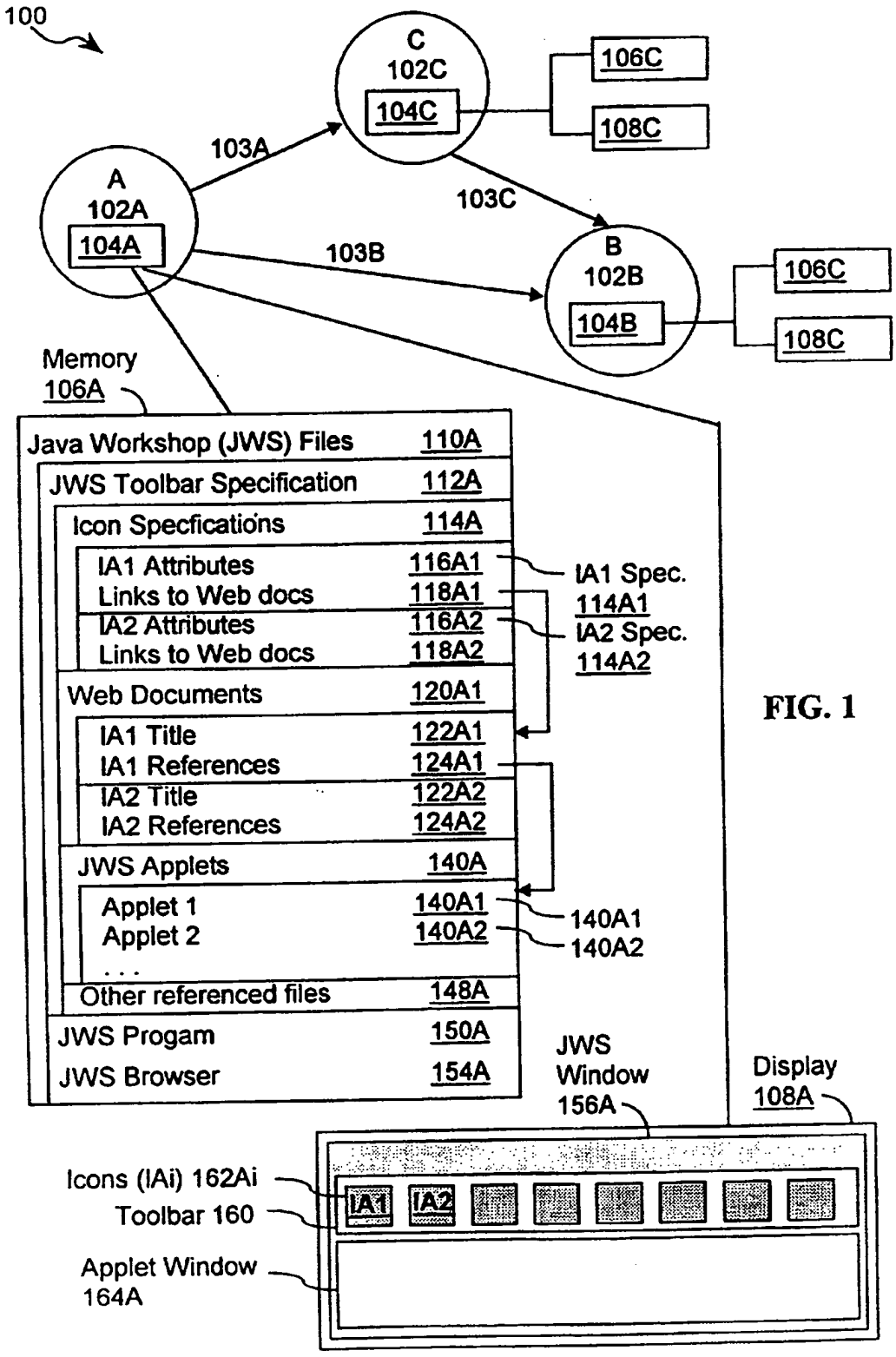
100
C
102C
104C
106C
108C
103A
103C
A
102A
104A
103B
B
102B
104B
106C
108C
Memory
106A
Java Workshop (JWS) Files 110A
JWS Toolbar Specification 112A
Icon Specfications 114A
IA1 Attributes 116A1
Links to Web docs 118A1
IA2 Attributes 116A2
Links to Web docs 118A2
IA1 Spec. 114A1
IA2 Spec. 114A2
Web Documents 120A1
IA1 Title 122A1
IA1 References 124A1
IA2 Title 122A2
IA2 References 124A2
JWS Applets 140A
Applet 1 140A1
Applet 2 140A2
. . .
140A1
140A2
Other referenced files 148A
JWS Progam 150A
JWS Browser 154A
JWS Window 156A
Display 108A
Icons (IAi) 162Ai
Toolbar 160
IA1
IA2
Applet Window 164A

FIG. 1

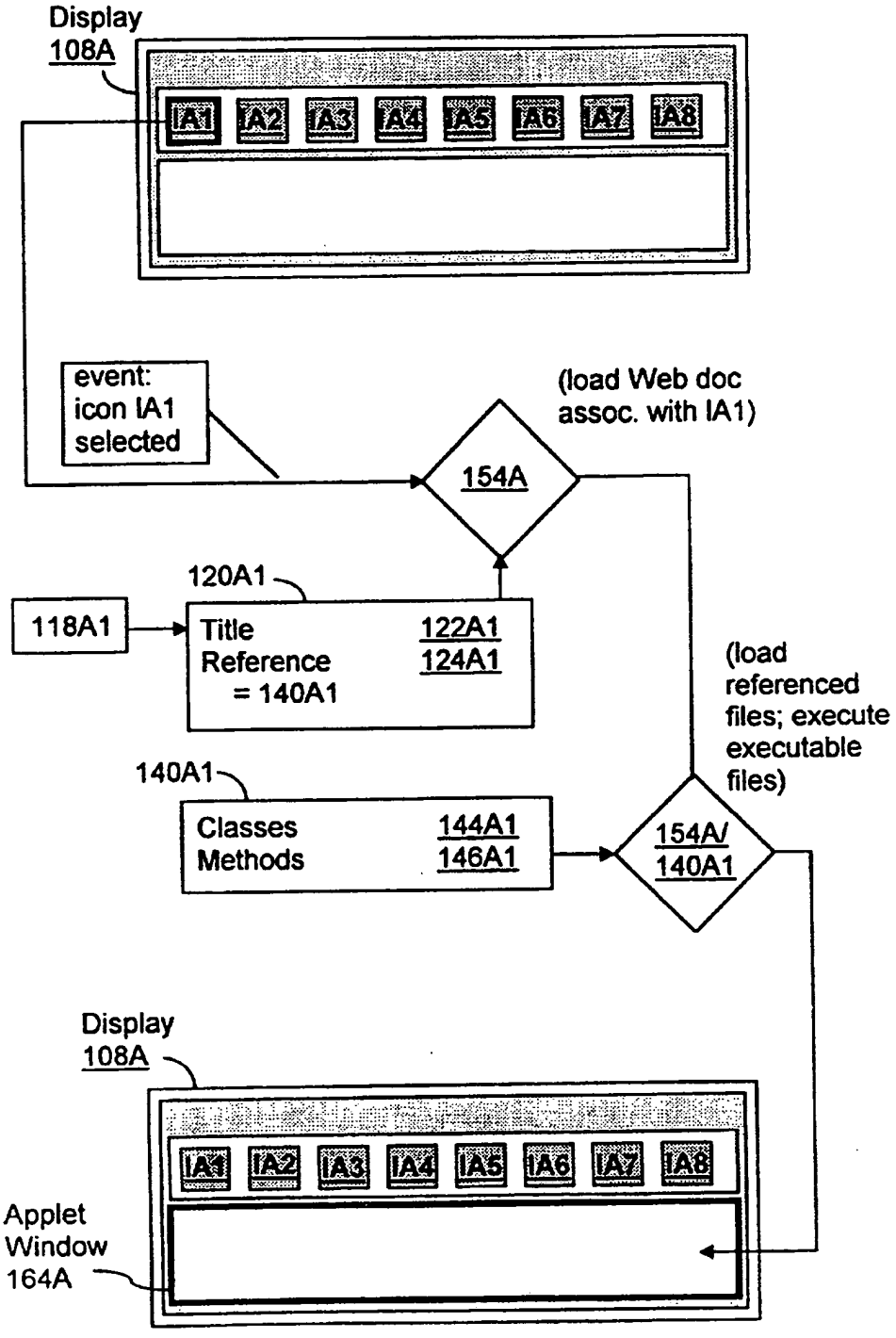
Display
108A
IA1
IA2
IA3
IA4
IA5
IA6
IA7
IA8
event:
icon IA1
selected
154A
(load Web doc
assoc. with IA1)
120A1
118A1
Title
Reference
= 140A1
122A1
124A1
(load
referenced
files; execute
executable
files)
140A1
Classes
Methods
144A1
146A1
154A/
140A1
Display
108A
IA1
IA2
IA3
IA4
IA5
IA6
IA7
IA8
Applet
Window
164A

FIG. 2

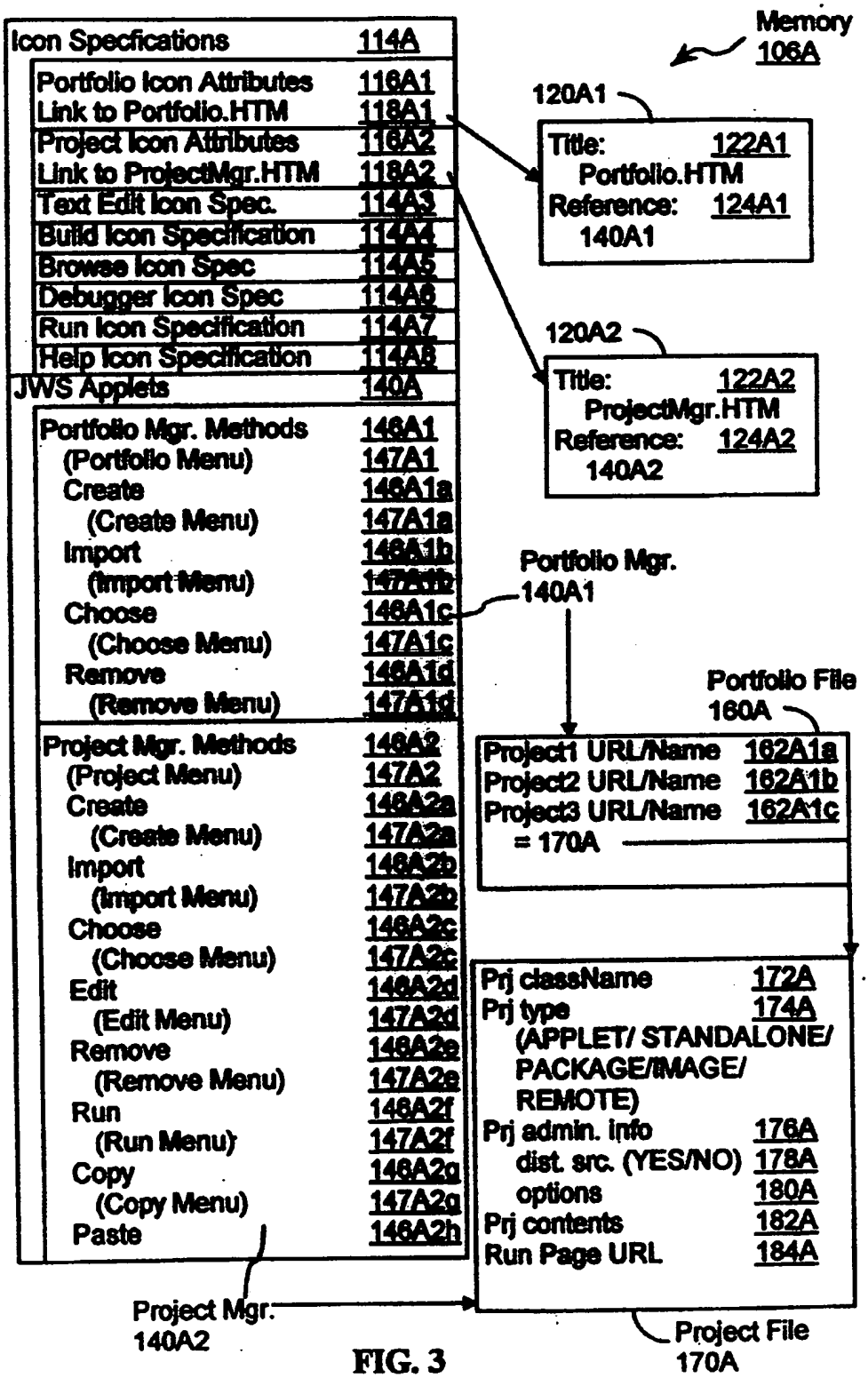
Memory
106A
Icon Specifications 114A
Portfolio Icon Attributes 116A1
Link to Portfolio.HTM 118A1
Project Icon Attributes 116A2
Link to ProjectMgr.HTM 118A2
Text Edit Icon Spec. 114A3
Build Icon Specification 114A4
Browse Icon Spec 114A5
Debugger Icon Spec 114A6
Run Icon Specification 114A7
Help Icon Specification 114A8
JWS Applets 140A
Portfolio Mgr. Methods 146A1
(Portfolio Menu) 147A1
Create 146A1a
(Create Menu) 147A1a
Import 146A1b
(Import Menu) 147A1b
Choose 146A1c
(Choose Menu) 147A1c
Remove 146A1d
(Remove Menu) 147A1d
Project Mgr. Methods 146A2
(Project Menu) 147A2
Create 146A2a
(Create Menu) 147A2a
Import 146A2b
(Import Menu) 147A2b
Choose 146A2c
(Choose Menu) 147A2c
Edit 146A2d
(Edit Menu) 147A2d
Remove 146A2e
(Remove Menu) 147A2e
Run 146A2f
(Run Menu) 147A2f
Copy 146A2g
(Copy Menu) 147A2g
Paste 146A2h
120A1
Title: 122A1
Portfolio.HTM
Reference: 124A1
140A1
120A2
Title: 122A2
ProjectMgr.HTM
Reference: 124A2
140A2
Portfolio Mgr.
140A1
Portfolio File
160A
Project1 URL/Name 162A1a
Project2 URL/Name 162A1b
Project3 URL/Name 162A1c
= 170A
Prj className 172A
Prj type 174A
(APPLET/ STANDALONE/
PACKAGE/IMAGE/
REMOTE)
Prj admin. info 176A
dist. src. (YES/NO) 178A
options 180A
Prj contents 182A
Run Page URL 184A
Project Mgr:
140A2
Project File
170A

FIG. 3

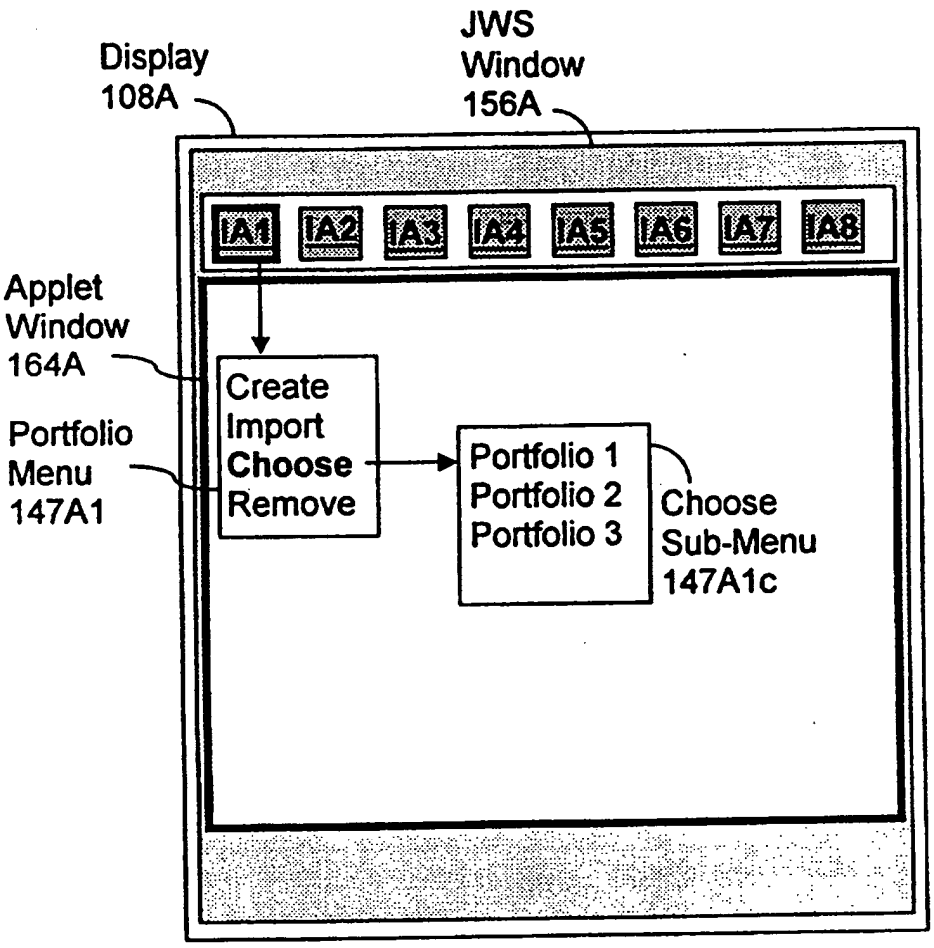
Display
108A
JWS
Window
156A
IA1
IA2
IA3
IA4
IA5
IA6
IA7
IA8
Applet
Window
164A
Portfolio
Menu
147A1
Create
Import
Choose
Remove
Portfolio 1
Portfolio 2
Portfolio 3
Choose
Sub-Menu
147A1c

FIG. 4

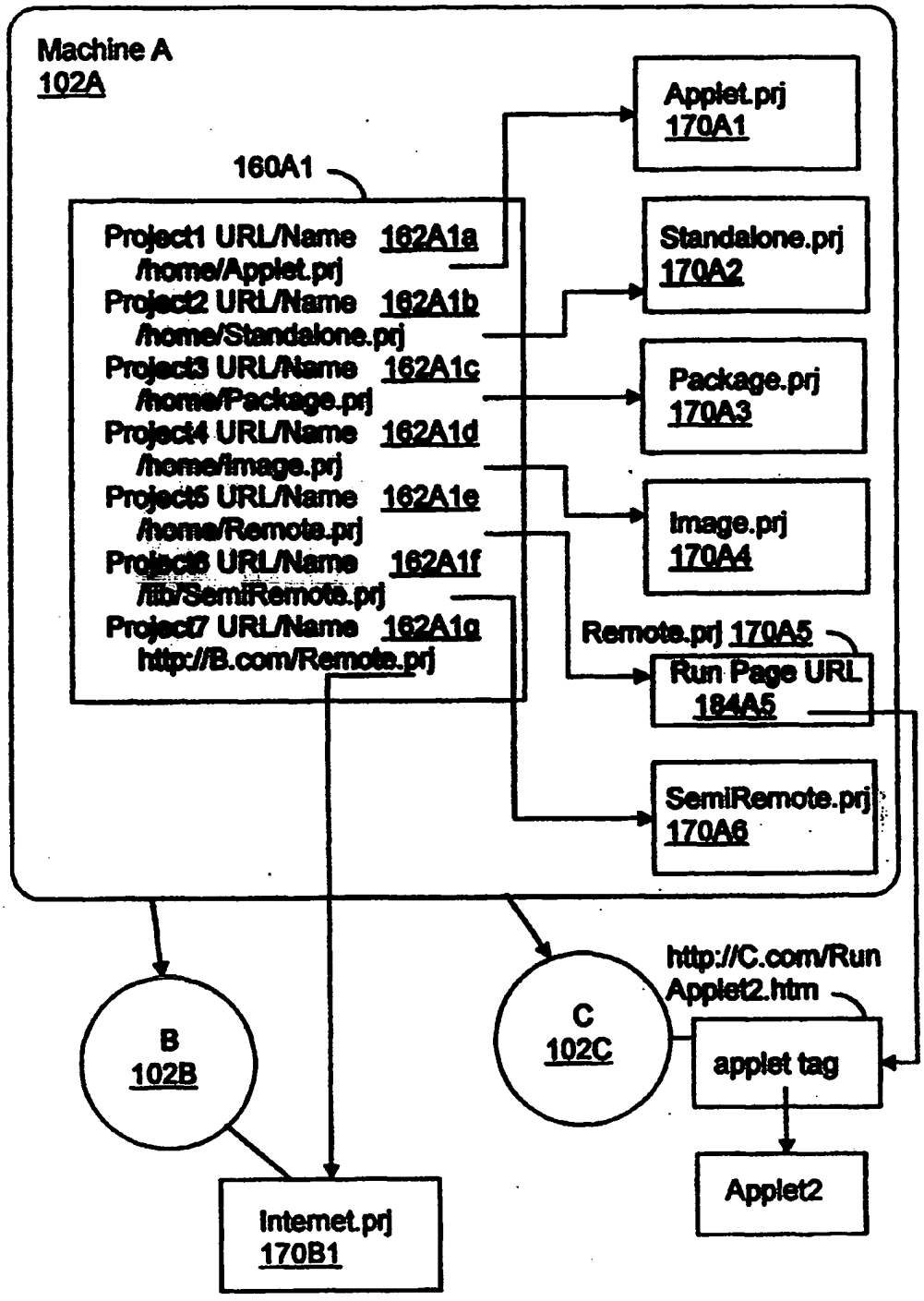
Machine A
102A
160A1
Project1 URL/Name 162A1a
/home/Applet.prj
Project2 URL/Name 162A1b
/home/Standalone.prj
Project3 URL/Name 162A1c
/home/Package.prj
Project4 URL/Name 162A1d
/home/Image.prj
Project5 URL/Name 162A1e
/home/Remote.prj
Project6 URL/Name 162A1f
/lib/SemiRemote.prj
Project7 URL/Name 162A1g
http://B.com/Remote.prj
Applet.prj
170A1
Standalone.prj
170A2
Package.prj
170A3
Image.prj
170A4
Remote.prj 170A5
Run Page URL
184A5
SemiRemote.prj
170A6
B
102B
C
102C
http://C.com/Run
Applet2.htm
applet tag
Applet2
Internet.prj
170B1

FIG. 5

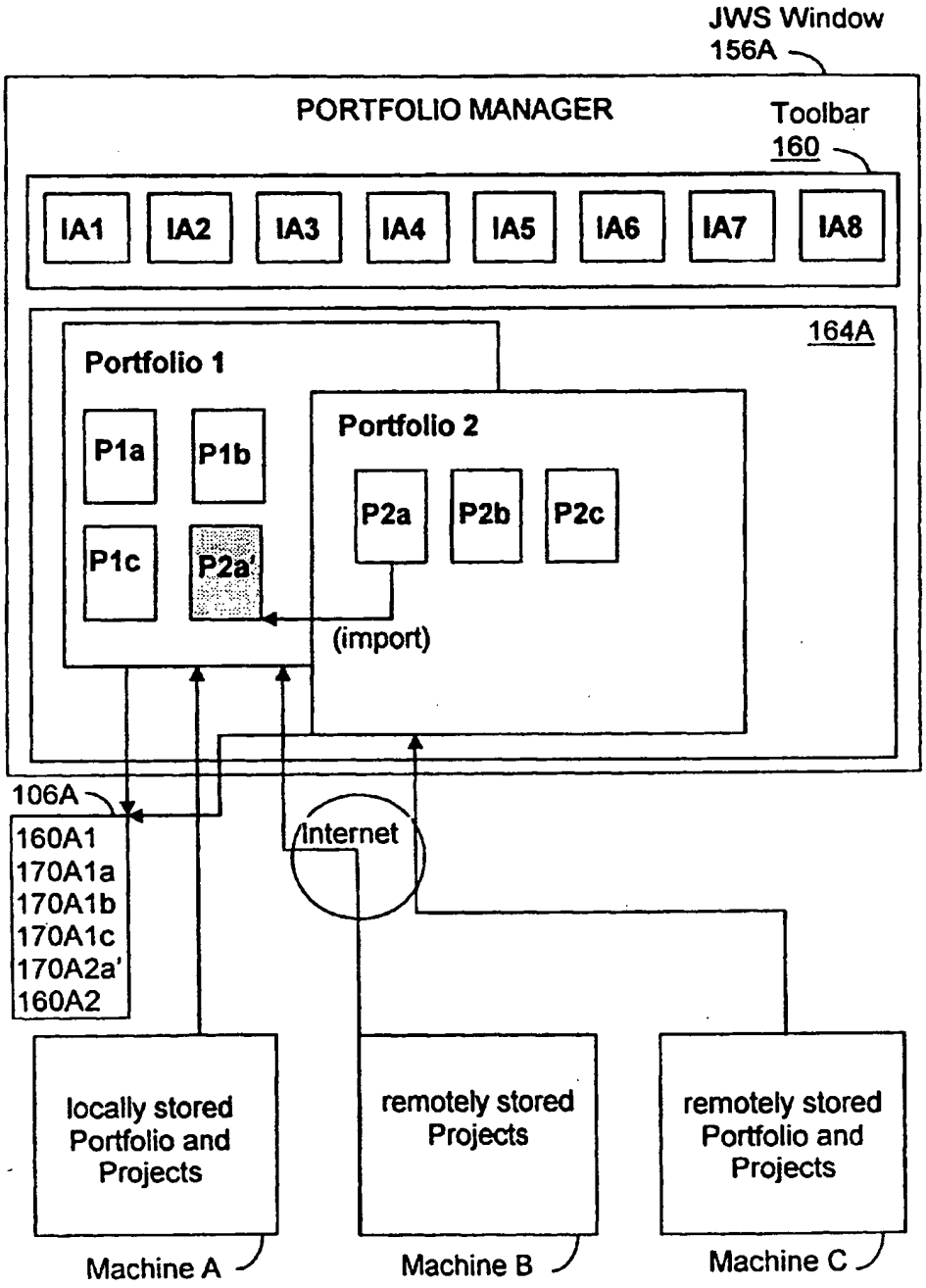
JWS Window
156A
PORTFOLIO MANAGER
Toolbar
160
IA1
IA2
IA3
IA4
IA5
IA6
IA7
IA8
164A
Portfolio 1
P1a
P1b
P1c
P2a'
Portfolio 2
P2a
P2b
P2c
(import)
106A
160A1
170A1a
170A1b
170A1c
170A2a'
160A2
Internet
locally stored Portfolio and Projects
remotely stored Projects
remotely stored Portfolio and Projects
Machine A
Machine B
Machine C

FIG. 6